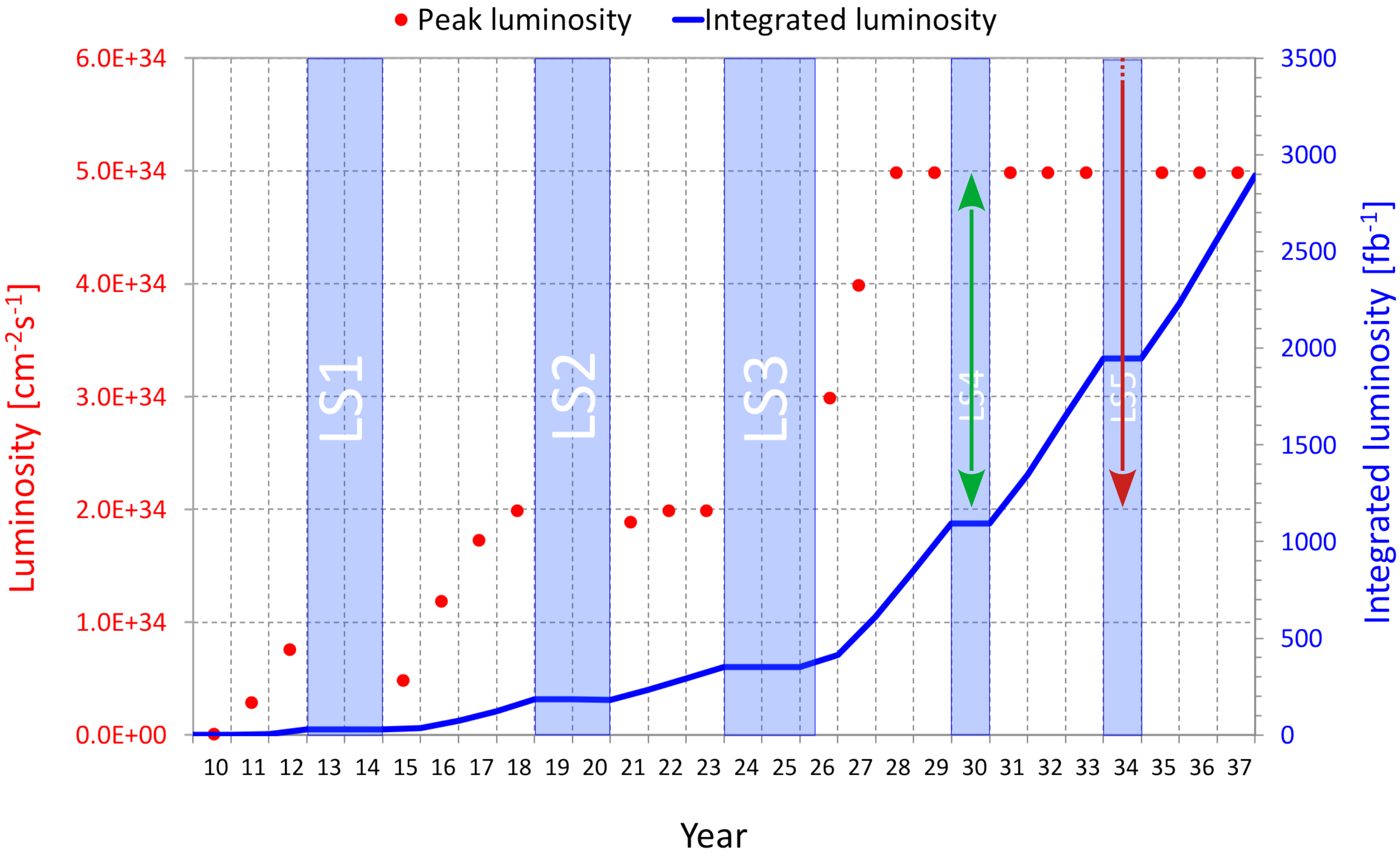
Towards a heterogeneous computing farm for the CMS High Level Trigger

Andrea Bocci[1], Vincenzo Innocente[1], Matti Kortelainen[2], Felice Pantaleo[1], Marco Rovere[1]

[1] CERN, [2] Fermilab

On behalf of the **CMS Collaboration**

the High Luminosity LHC



higher luminosity

→ 2.5 × ~ 4 ×

higher detector readout rate

→ 5 × ~ 7.5 ×

we have a problem !

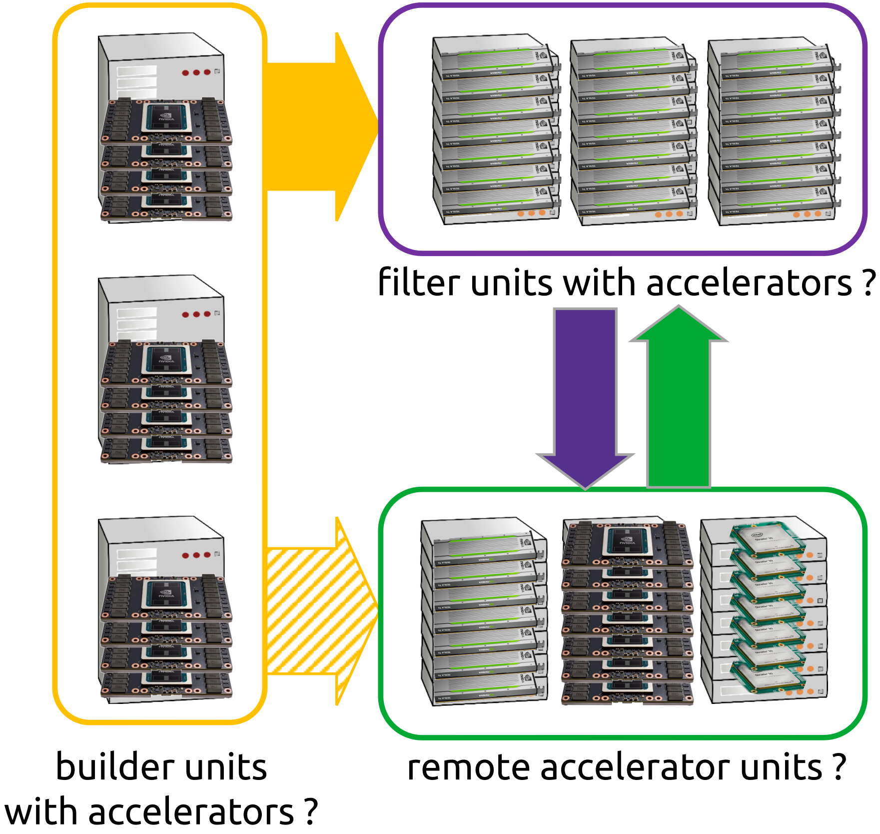
- higher pileup
  - optimistic extrapolation: × 3
- new detectors
  - based on MC: × 1.3
- higher L1 trigger rate
  - design: **× 7.5**

- “Phase 2” HLT resource needs
  - optimistically … × 30

- expected improvements in CPUs
  - optimistic: × 4
  - realistic: × 2

looking ahead

- get production ready
  - towards a possible deployment in Run 3
- integrate in the “official” CMSSW releases
  - CUDA-based framework
  - pixel local reconstruction, track and vertices
  - calorimeters’ local reconstruction
  - full tracking
- investigate different topologies
  - and communication / offload models
- a new programming model
  - GPU/CPU code sharing and reuse



Patatrack:

Accelerated Pixel Track reconstruction in CMS

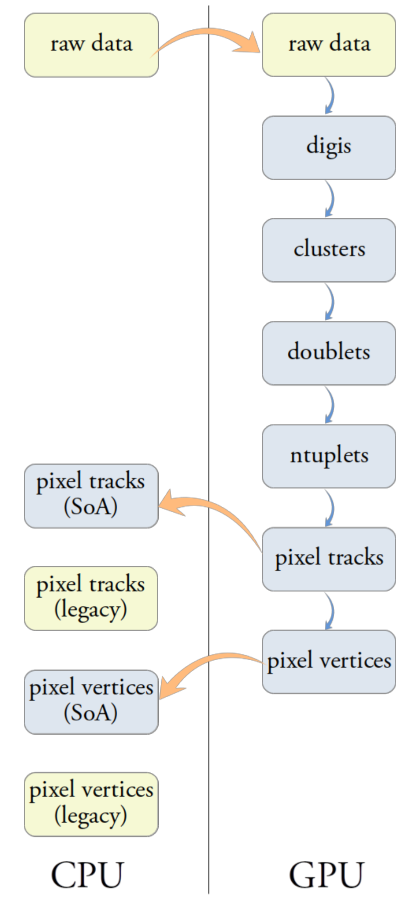
Andrea Bocci[1], Vincenzo Innocente[1], Matti Kortelainen[2], Felice Pantaleo[1], Roberto Ribatti[3], Marco Rovere[1], Gimmy Tomaselli[3]

[1]CERN – Experimental Physics Department, [2]FNAL, [3]Scuola Normale Superiore di Pisa

felice@cern.ch

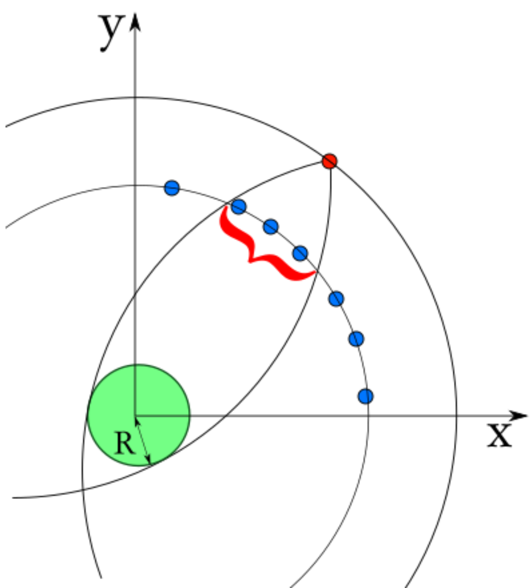
Patatrack Pixel Reconstruction Workflow

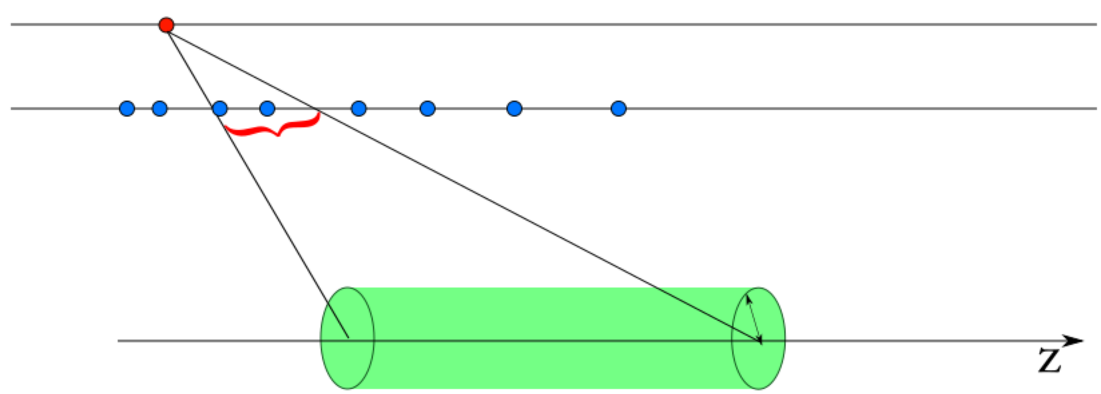
- Full Pixel Track reconstruction in CMSSW
  - from Raw data decoding to Primary Vertices determination
- Raw data for each event is transferred to the GPU initially (~250kB/event)
- At each step data can be transferred to CPU and used to populate "legacy" event data
- The standard validation is fully supported
- Integer results are identical
- Small differences in the results of floating point can be explained by differences in re-association



Doublets

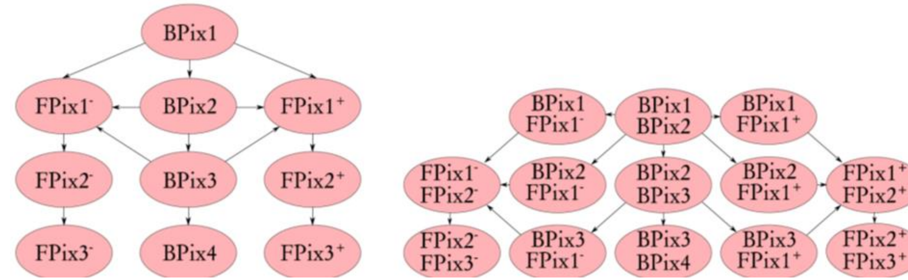
- The local reconstruction produces hits
- Doublets are created opening a window depending on the tracking region/beamspot and layer-pair
    - The cluster size along the beamline can be required to exceed a minimum value for barrel hits connecting to an endcap layer
- Hits within the bins are connected to form doublets if they pass further "alignment cuts" based on their actual position
- In the barrel the compatibility of the cluster size along the beamline between the two hits can be required
- The cuts above reduce the number of doublets by an order of magnitude and the combinatorics by a factor 50





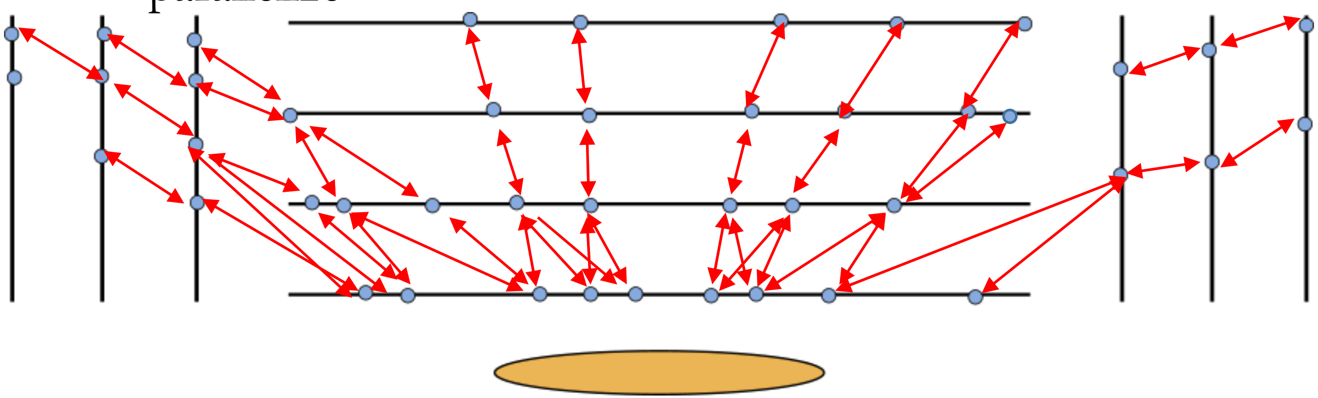
Cellular Automaton-based Hit Chain-Maker

The CA is a track seeding algorithm designed for parallel architectures



It requires a list of layers and their pairings

- A graph of all the possible connections between layers is created
- Doublets aka Cells are created for each pair of layers, in parallel at the same time
- Fast computation of the compatibility between two connected cells, in parallel
- No knowledge of the world outside adjacent neighboring cells required, making it easy to parallelize

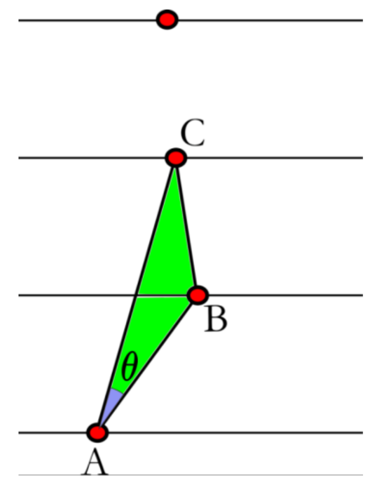


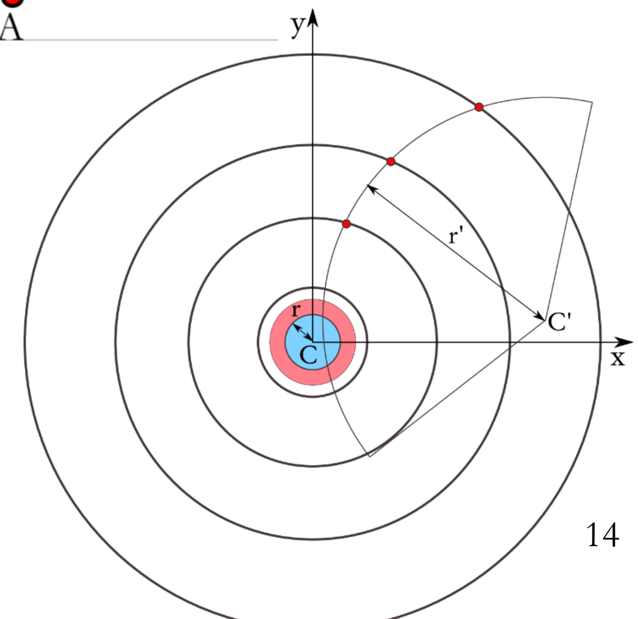
- Better efficiency and fake rejection wrt previous algo
- Since 2017 data-taking has become the default track seeding algorithm for all the pixel-seeded online and offline iterations

- In the following, at least four hits are required, but triplets can be kept to recover efficiency where geometric acceptance lacks one hit

CA compatibility cuts

- The compatibility between two cells is checked only if they share one hit
  - AB and BC share hit B
- In the R-z plane a requirement is alignment of the two cells
- In the cross plane the compatibility with the beamspot region





Fits

Pixel track "fit" at the HLT is still using 3 points for quadruplets and errors on parameters are loaded from a look-up table[eta][pT]

The Patatrack Pixel reconstruction includes two Multiple Scattering-aware fits:

- Riemann Fit
- Broken Line Fit

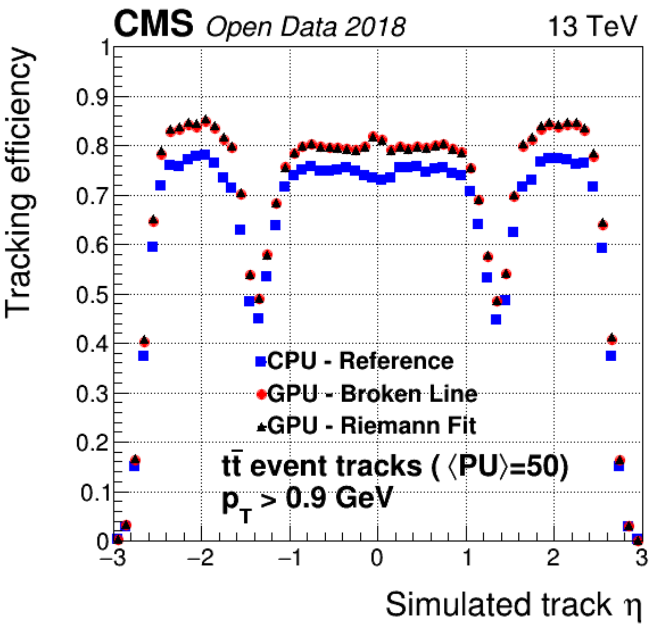
They allow to better exploit information coming from our 4-layer pixel detector and improve parameter resolutions and fake rejection

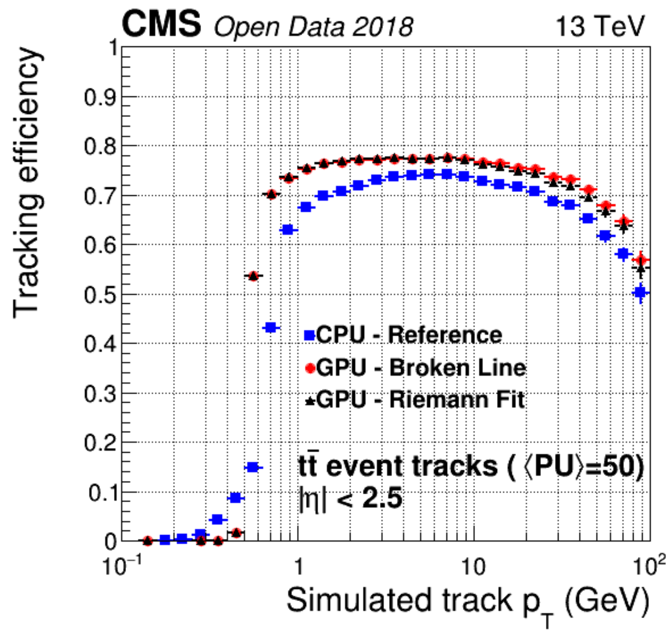
Performance Definitions

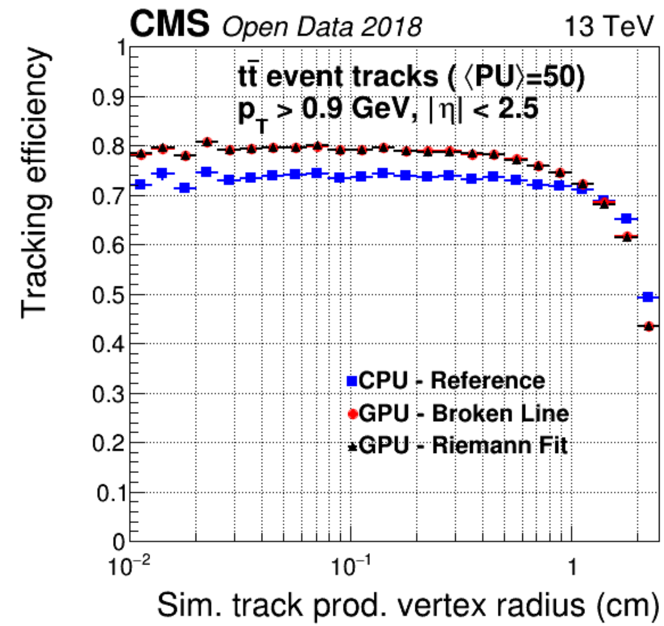
Physics performance:

- 20000 MC ttbar events <PU> = 50, design conditions, 25ns, sqrt(s)=13TeV
- Matching of reconstructed tracks with simulated ones requires that all hits of the reconstructed track come from the same simulated track
- Efficiency: number of matched reconstructed tracks divided by number of simulated tracks
- Fake Rate: number of non-matched reconstructed tracks divided by number of reconstructed tracks
- Efficiency is computed only with respect to the hard scatter.
- Efficiency has the following implicit cut: $|d_0| < 3.5$ cm additionally to the cuts quoted in the plots
- Duplicate is a reconstructed track matching to a simulated track that itself is matched to >= 2 tracks

Physics Performance - Efficiency

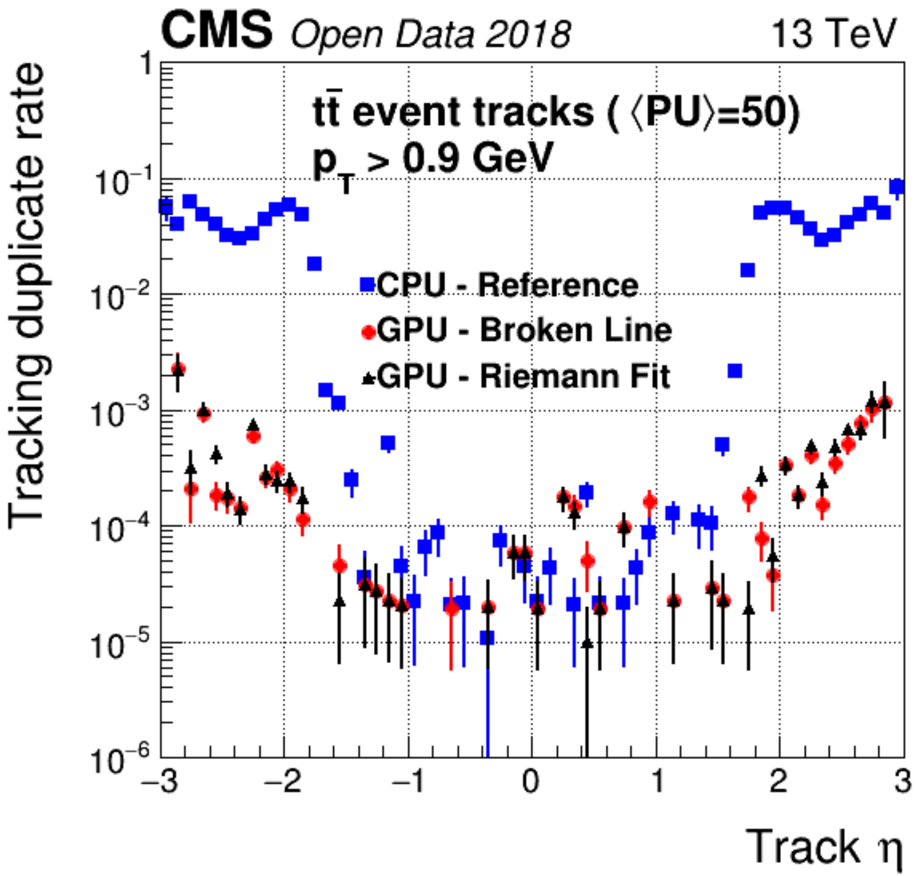


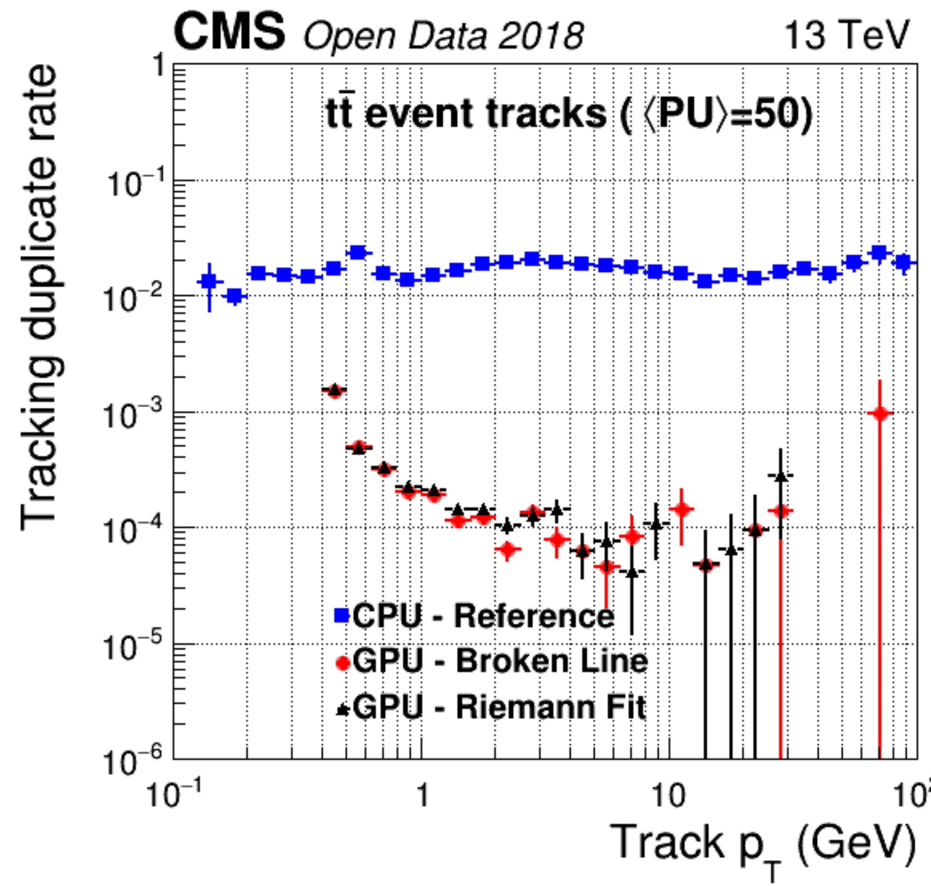




Track reconstruction efficiency as a function of simulated track η, p_T, and production vertex radius.

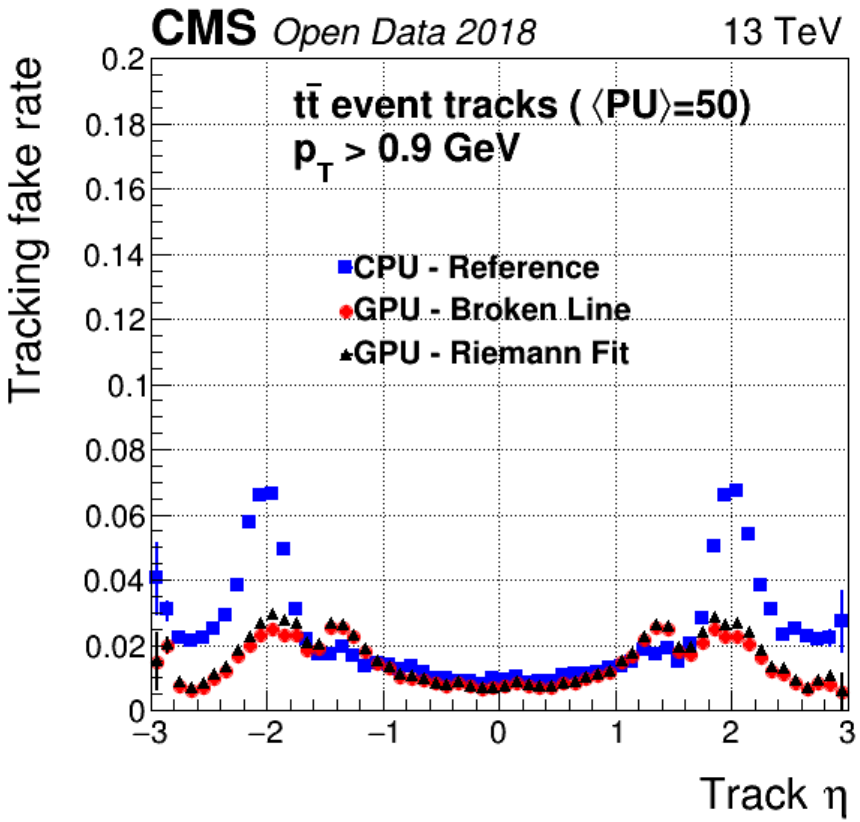
Physics performance - Duplicates

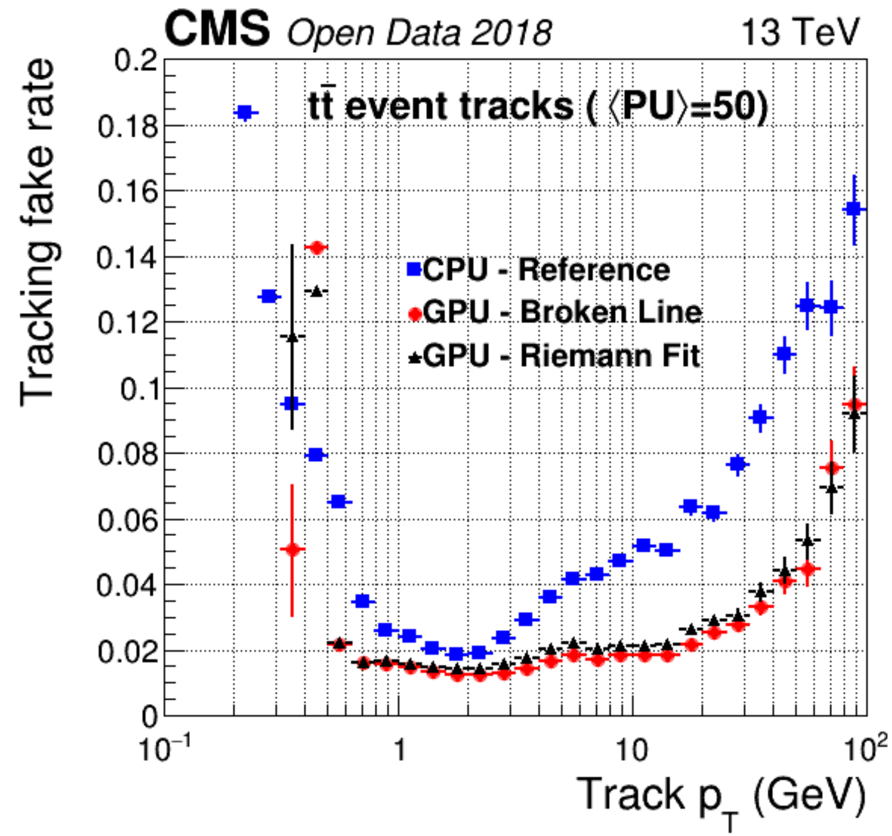




Track reconstruction duplicate rate as a function of reconstructed tracks η, p_T

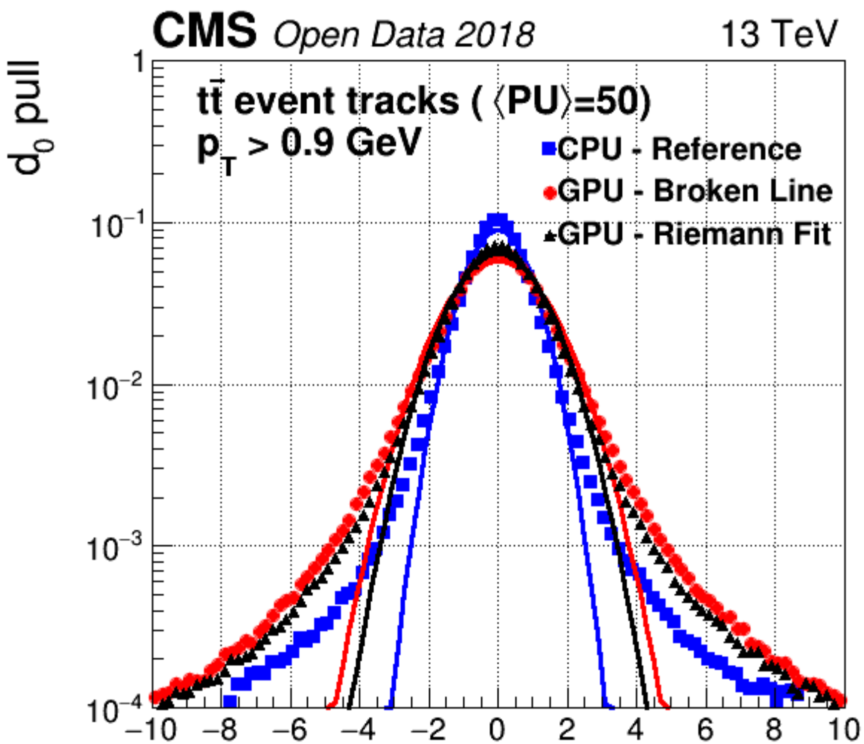
Physics performance – Fakes

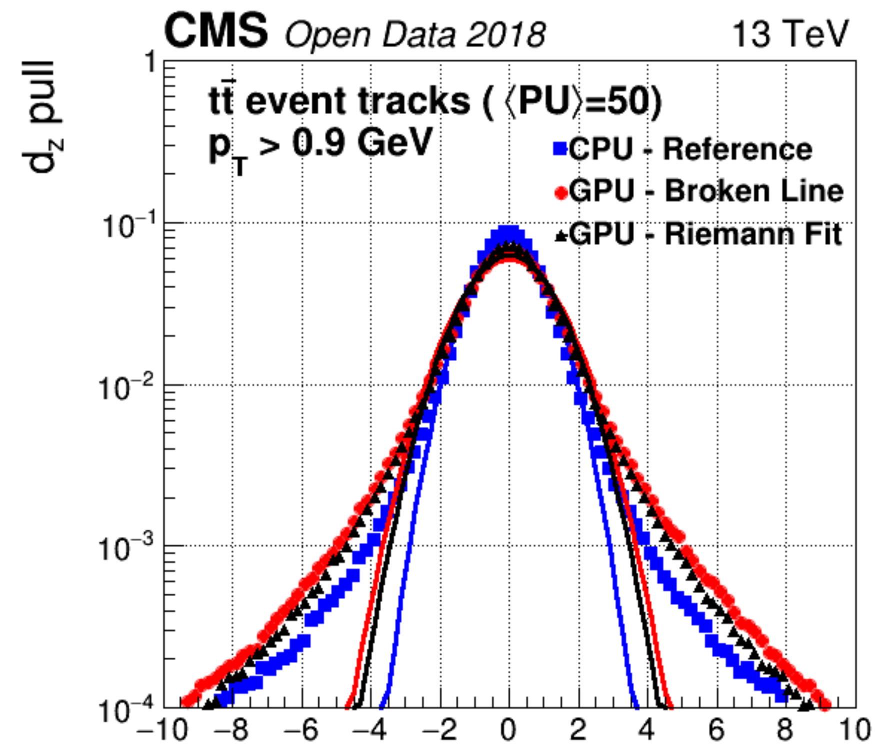




Track reconstruction fake rate as a function of reconstructed tracks η, p_T

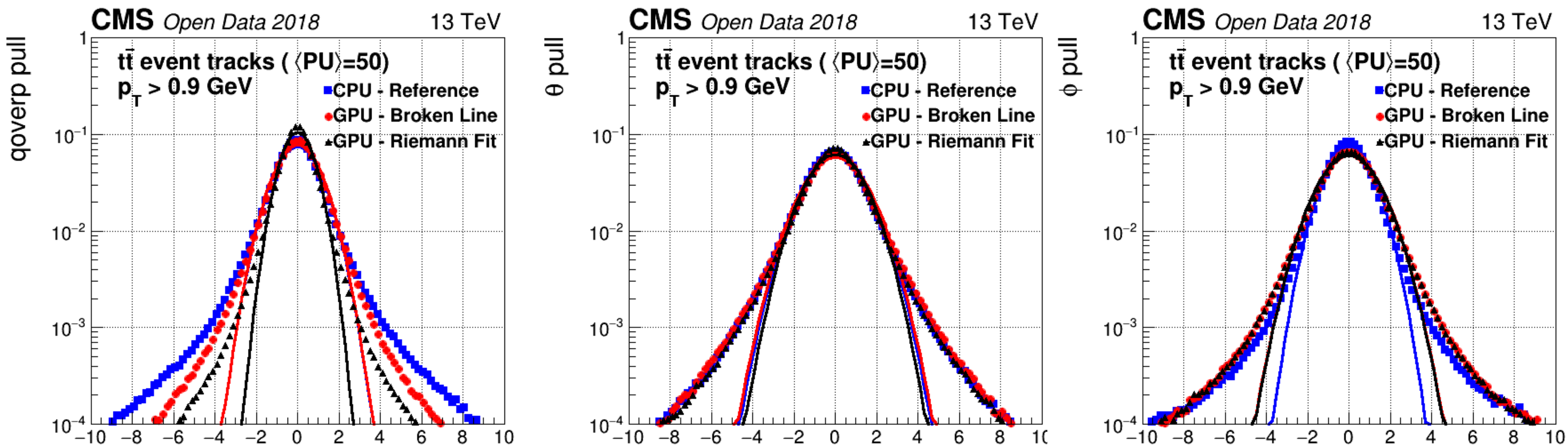
Physics Performance – Fit Pulls





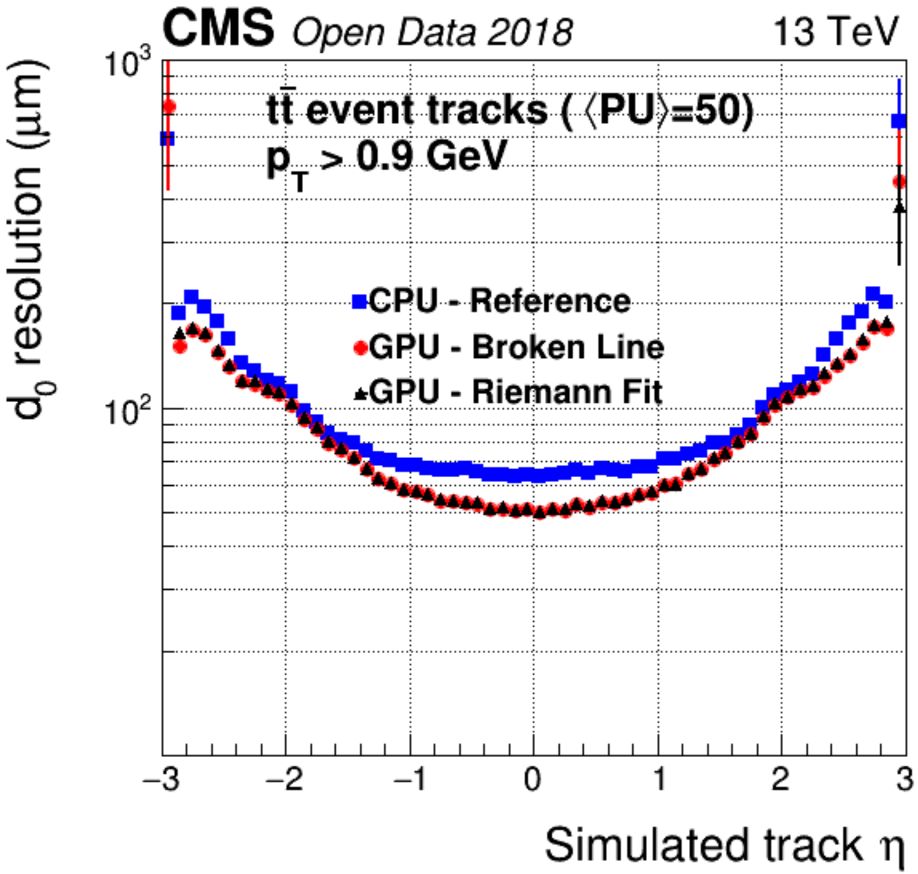
	σ - Reference	σ - Broken Line	σ - Riemann Fit
d_0	0.84	1.32	1.18
d_z	0.97	1.28	1.20

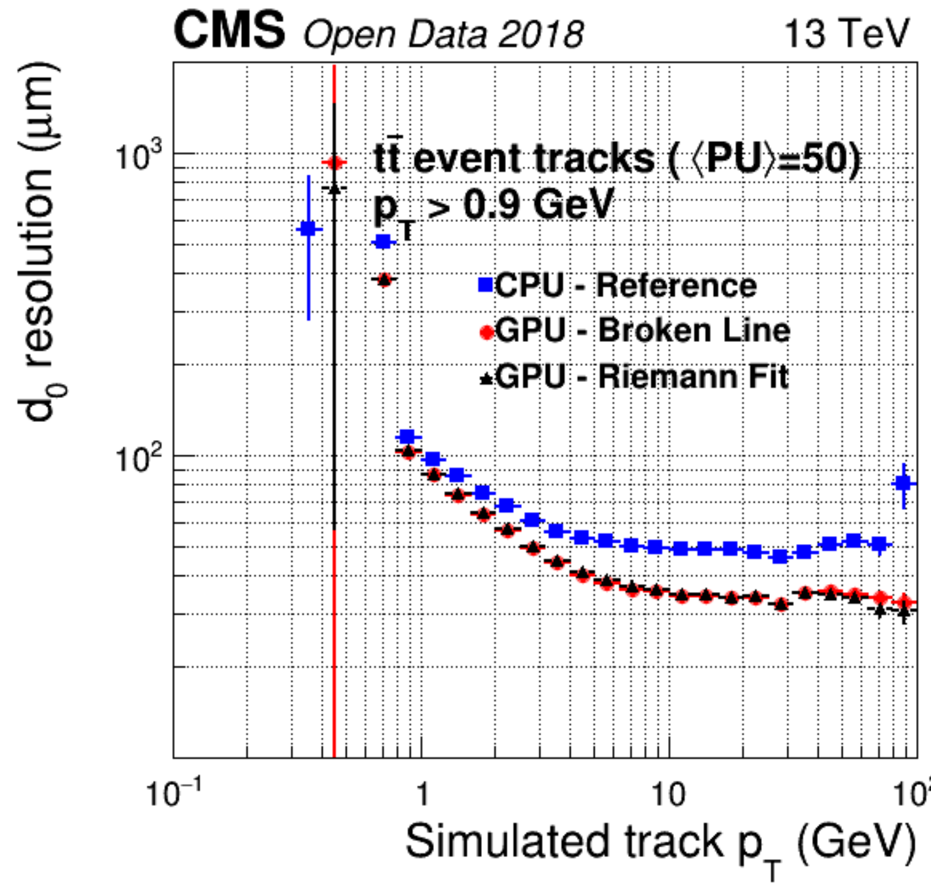
Physics Performance – Fit Pulls



	σ - Reference	σ - Broken Line	σ - Riemann Fit
qoverp	0.99	0.99	0.72
θ	1.29	1.33	1.22
φ	1.02	1.28	1.27

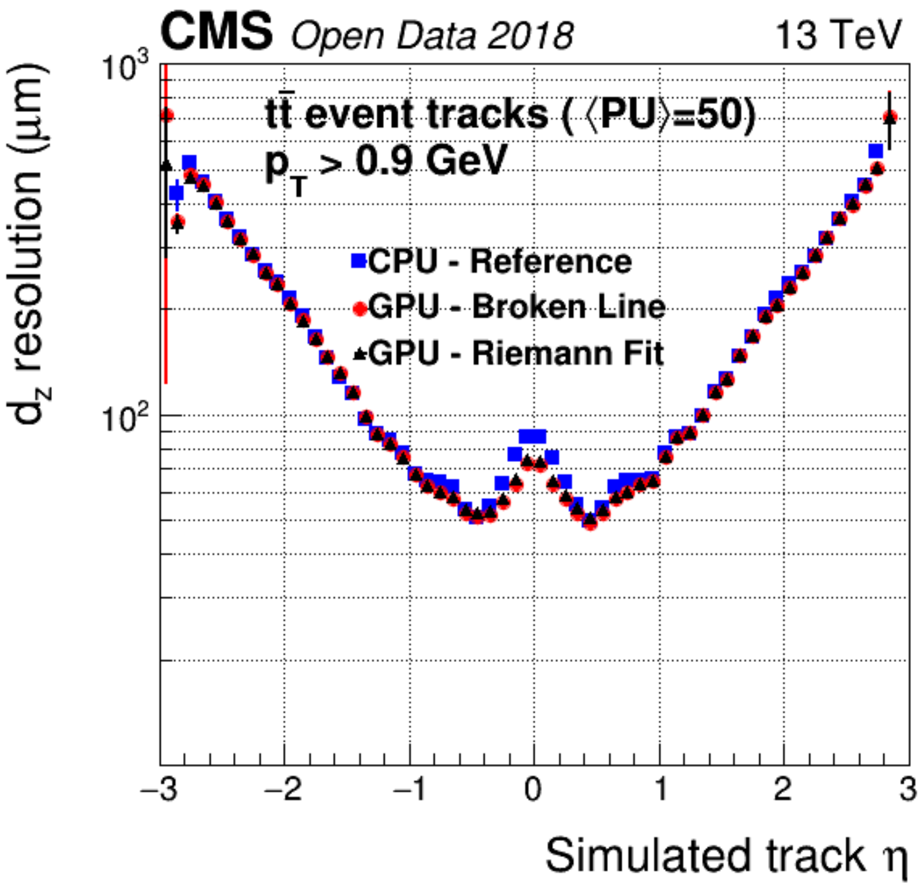
Physics Performance - Resolutions

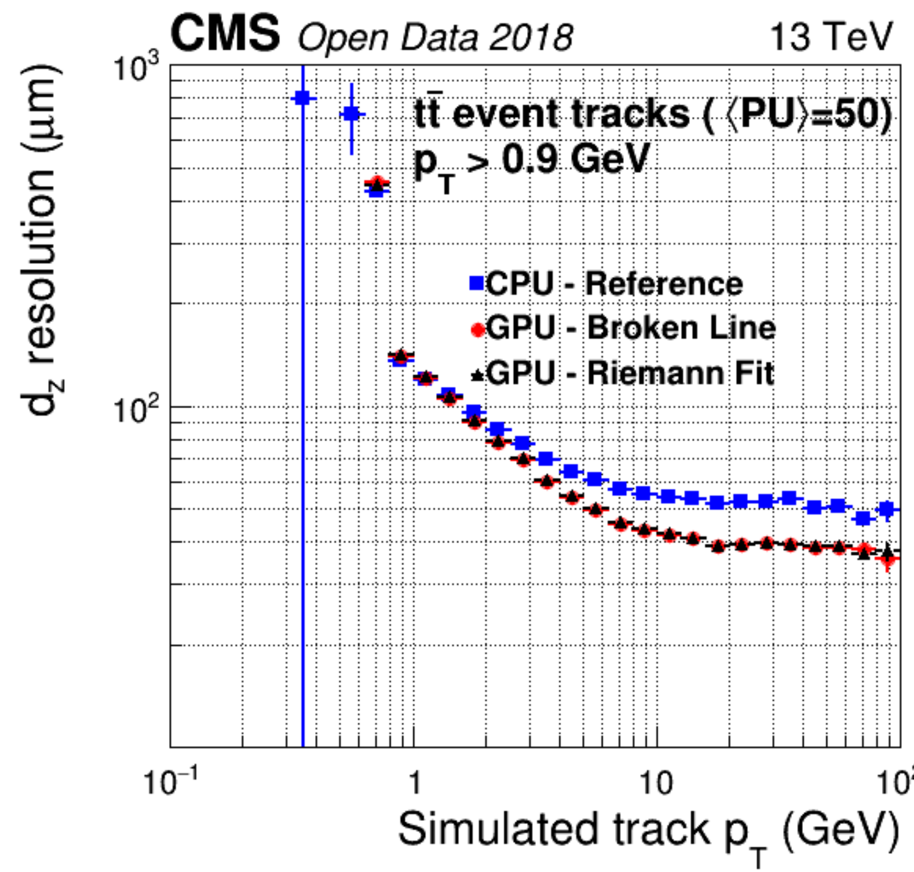




Track resolution of the transverse impact parameter as a function of simulated track η and p_T

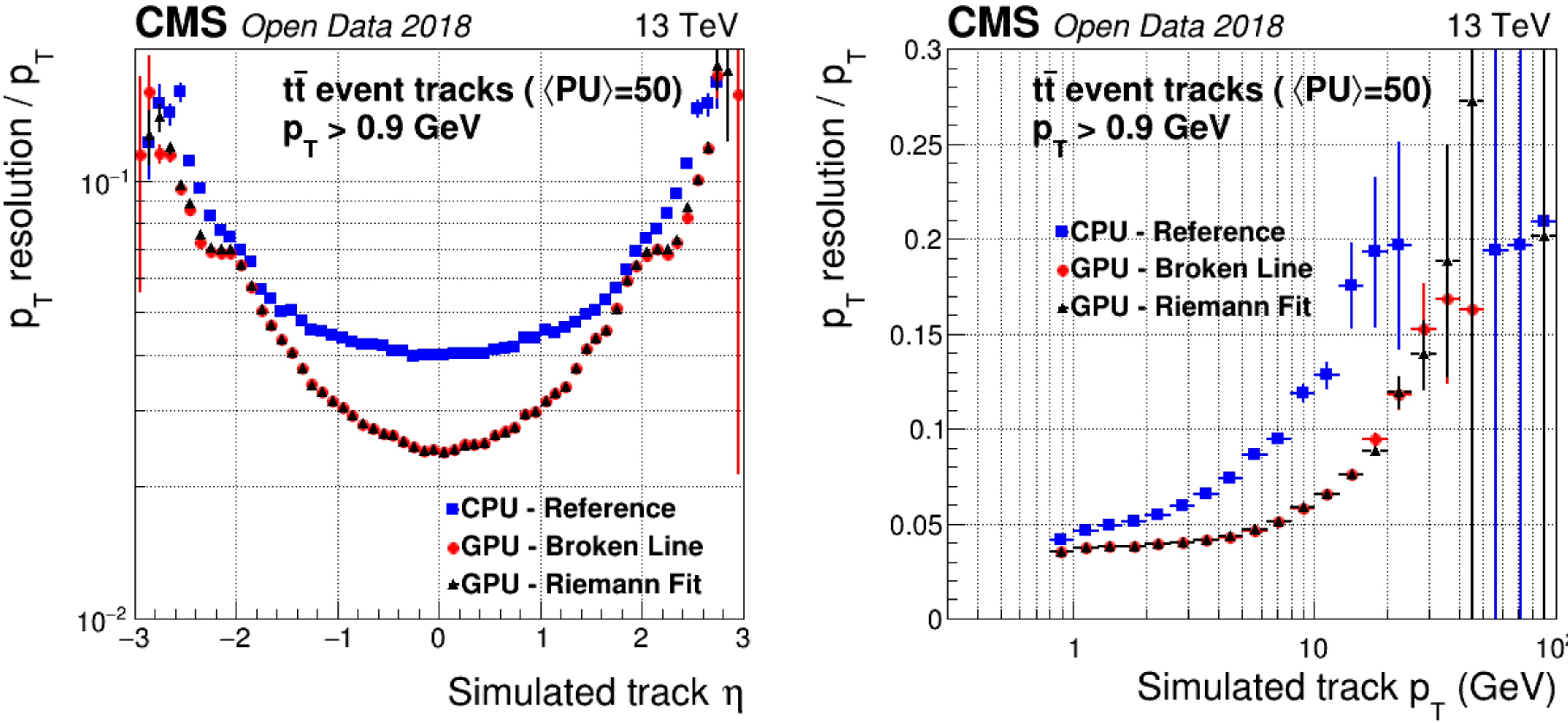
Physics Performance - Resolutions





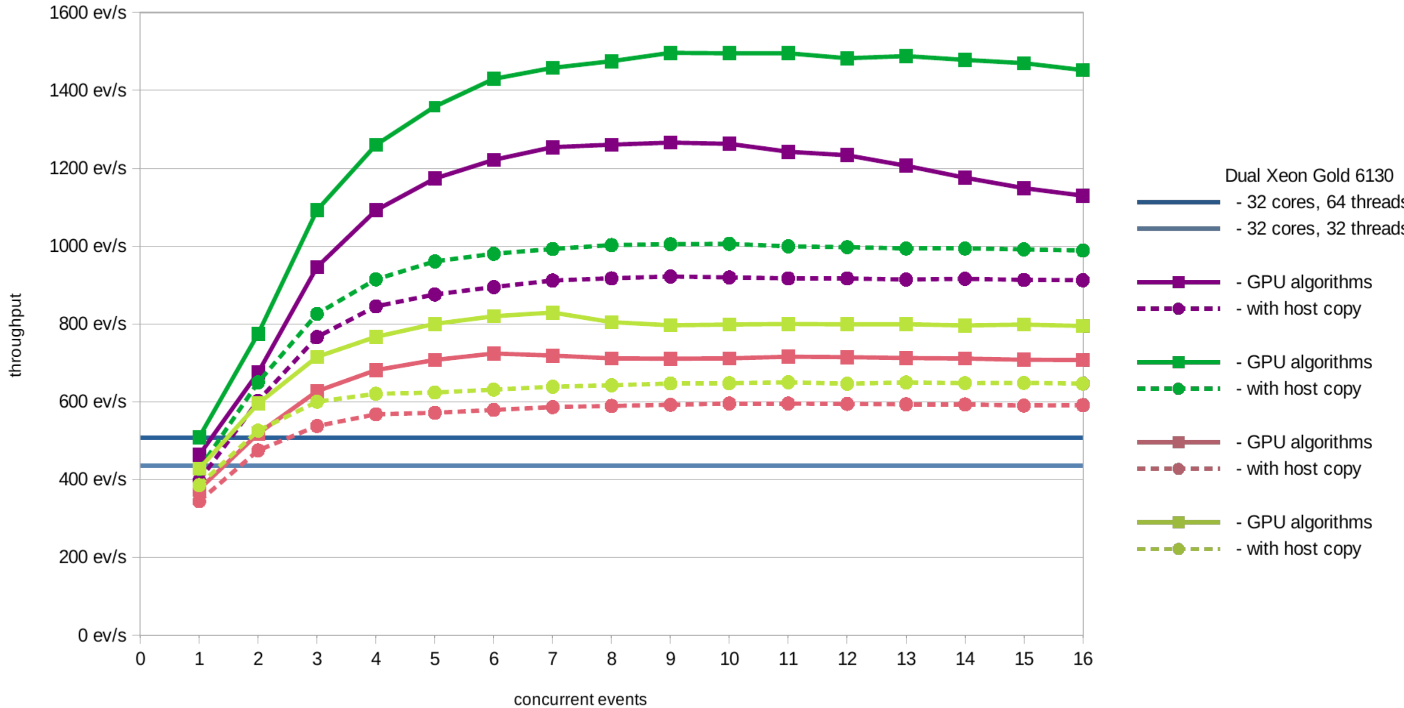
Track resolution of the longitudinal impact parameter as a function of simulated track η and p_T

Physics Performance - Resolutions



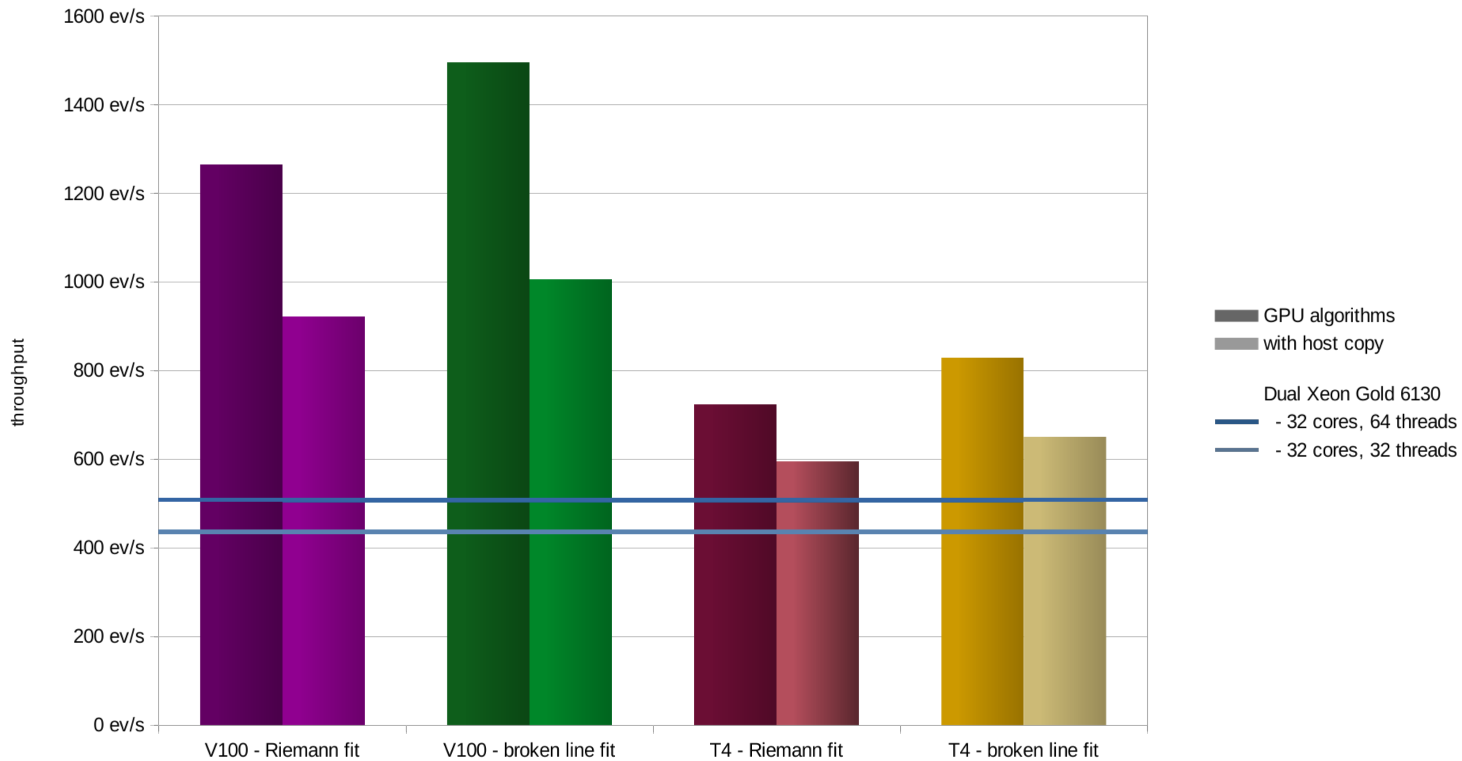
Track reconstruction resolution of p_T as a function of simulated track η and p_T

Computational Performance



Pixel reconstruction consumers can either work directly on the GPU or ask for a copy of the tracks and vertices on the host

Computational Performance



Pixel reconstruction consumers can either work directly on the GPU or ask for a copy of the tracks and vertices on the host

http://trackreco.github.io/
http://computing.fnal.gov/hepreco-scidac4/

mkFit Project: Speeding up particle track reconstruction using a vectorized and parallelized Kalman Filter algorithm

G. Cerati[1], P. Elmer[3], B. Gravelle[5], M. Kortelainen[1], S. Krutelyov[4], S. Lantz[2], M. Masciovecchio[4], K. McDermott[2], B. Norris[5], **A. Reinsvold Hall[1]**, D. Riley[2], M. Tadel[4], P. Wittich[2], F. Würthwein[4], A. Yagil[4]

1. **FNAL** 2. Cornell 3. Princeton 4. UCSD 5. Oregon

Parallelized KF Tracking Project

- Ongoing project for 3+ years
- **Mission:** adapt traditional Kalman Filter (KF) tracking algorithms to maximize usage of **vector units** and **multicore** architectures
  - Testing on Intel Xeon and Intel Xeon Phi
  - Longer term: adapt algorithm for GPUs (not covered today)
- **Achievements shown today:**
  - Effective use of vectorization and multi-thread scaling
  - Physics performance comparable to offline CMS reconstruction
- **Aim:** Test algorithm online in Run 3 software-based High Level Trigger (HLT), extend to HL-LHC CMS geometry

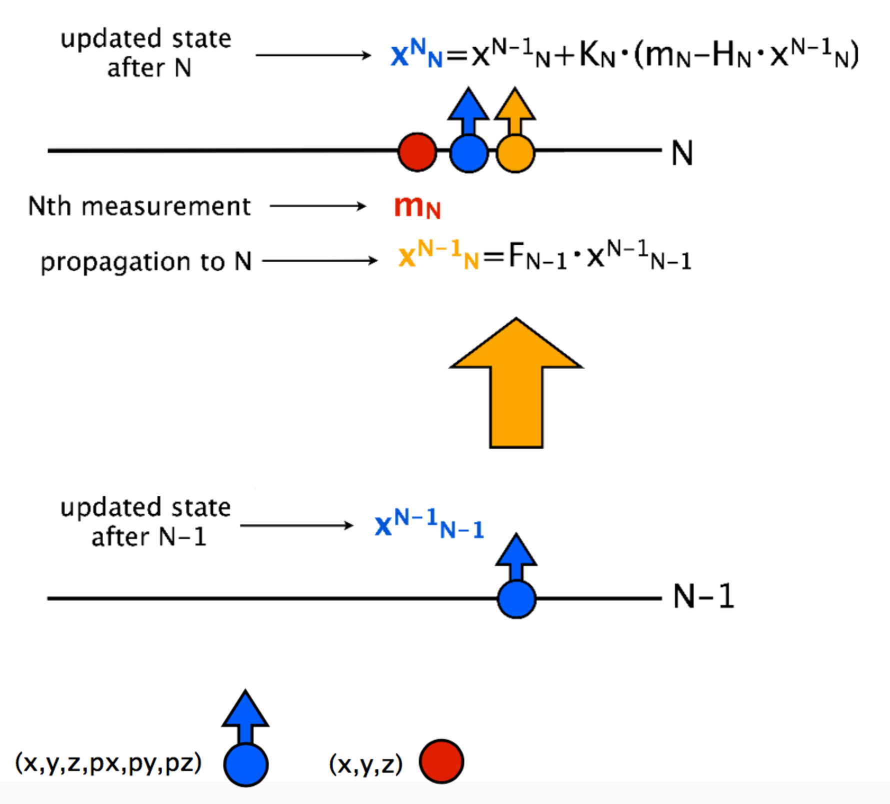
See project website for details:
http://trackreco.github.io/

Using the Kalman Filter

Benefits of the Kalman Filter for track finding/fitting:

- Robust handling of multiple scattering, energy loss, and other material effects
- Widely used in HEP
- Demonstrated physics performance

Predicted track state
Detector measurement
Updated track state

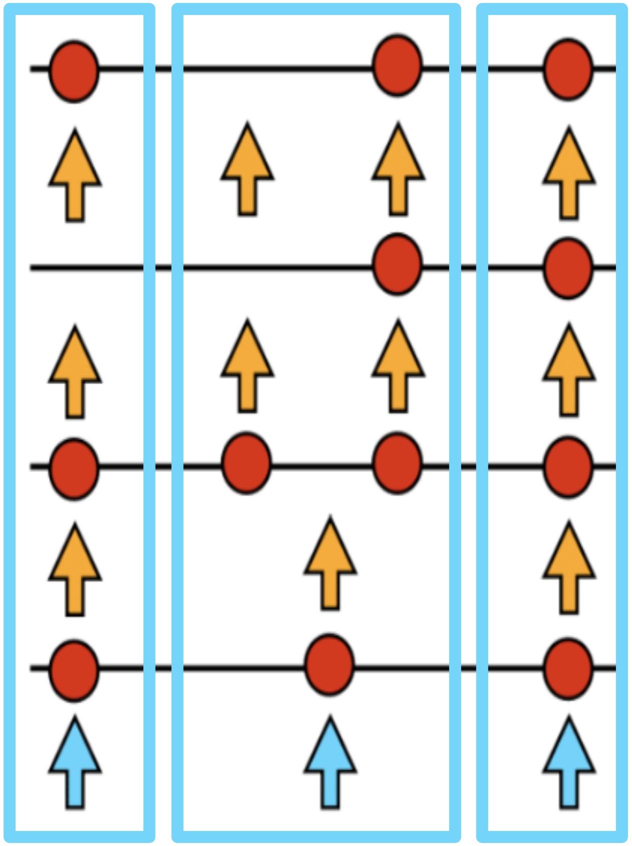


Two step process:

1. **Propagate** the track state from layer N-1 to layer N (prediction)
2. **Update** the state using the detector hit (measurement) on layer N

Track building in a nutshell

- Start with a **seed track**
- **Propagate** track state to the next detector layer
- Find detector hits near projected intersection of track with layer
  - Problem of **combinatorics**: could find 0 hits, 1 hit, or several hits
- Select best fit track-hit combinations as track candidates
  - Update estimated state of all track candidates with new hit
  - At each layer, the number of possible track candidates per seed increases
- **Iterate**

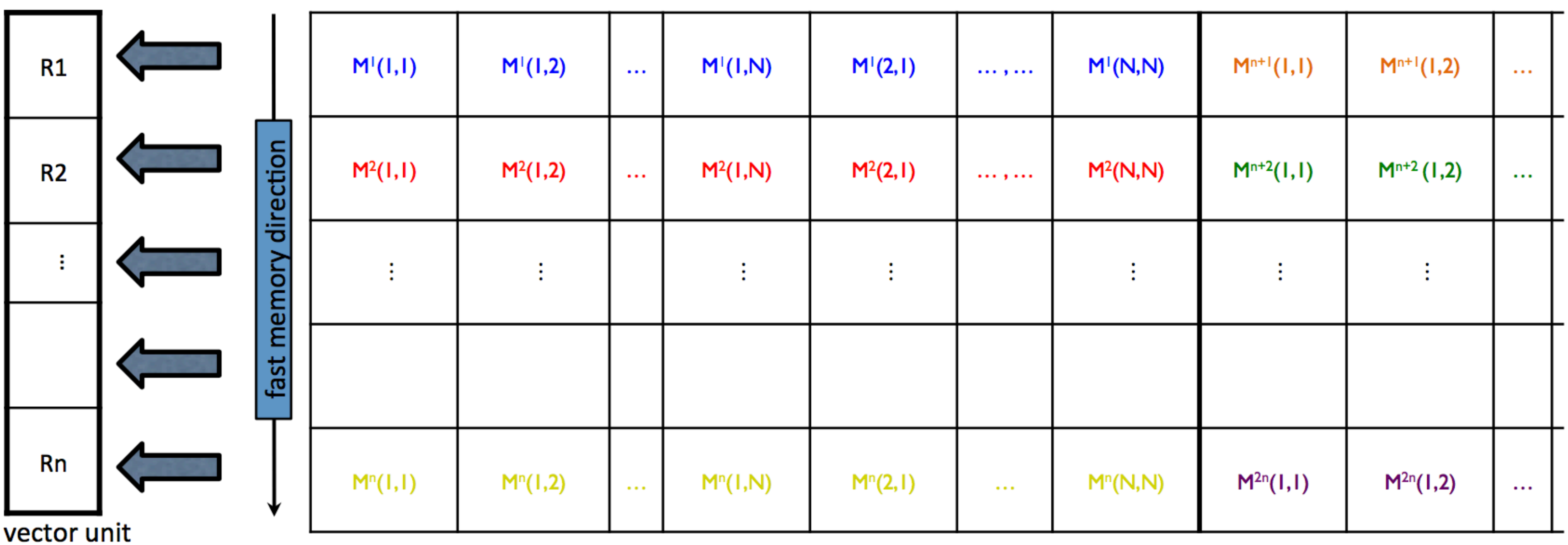


Parallelization and Vectorization

- **Task** scheduling is handled via TBB library from Intel
- **Parallelization** at multiple levels
  - **parallel for:** N events in flight
    - **parallel for:** 5 regions in η in each event
      - **parallel for:** seed-driven batching, 16 or 32 seeds per batch
- **Vectorized** processing of individual track candidates where possible
  - Using both compiler vectorization and the Matriplex library

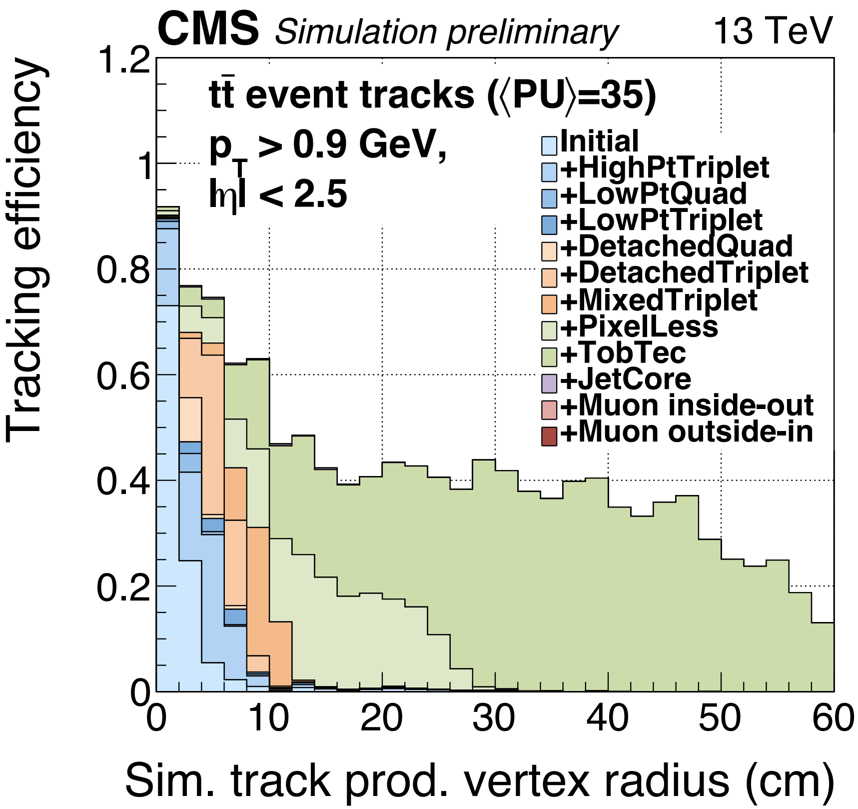
Matriplex Library

- Custom library for vectorization of small matrix operations
- "Matrix-major" representation designed to fill a vector unit with **n** small matrices and operate on each matrix in sync
- Includes code generator to generate C++ code or intrinsics for matrix multiplication of a given dimension
  - Can be told about known 0 and 1 elements matrices to reduce number of operations by up to 40%
- Used for all Kalman filter related operations



Matrix size **NxN**, vector unit size **n**

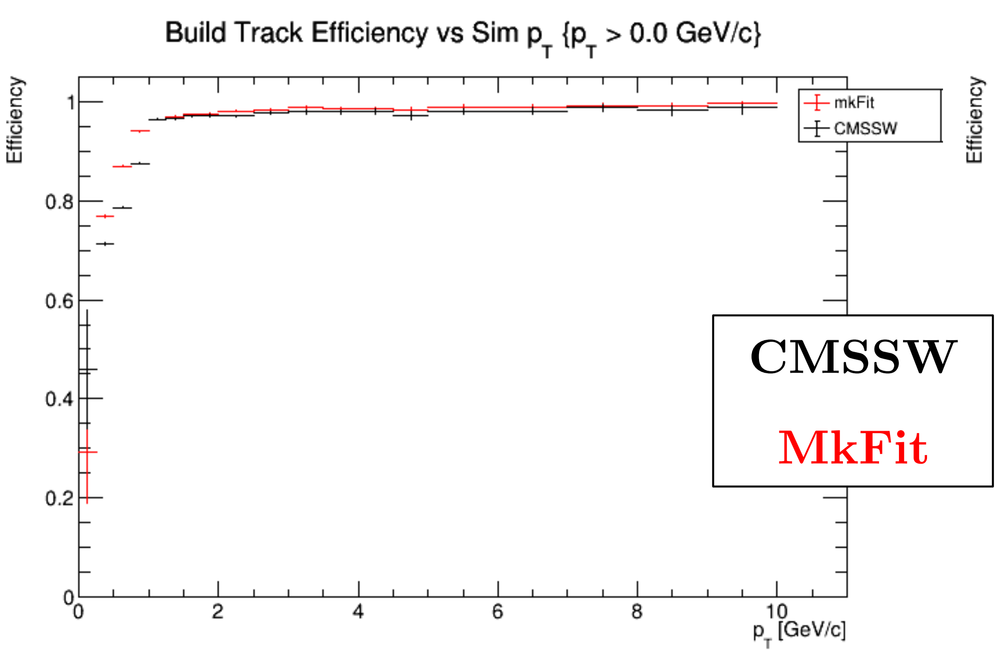
CMS Iterative Tracking

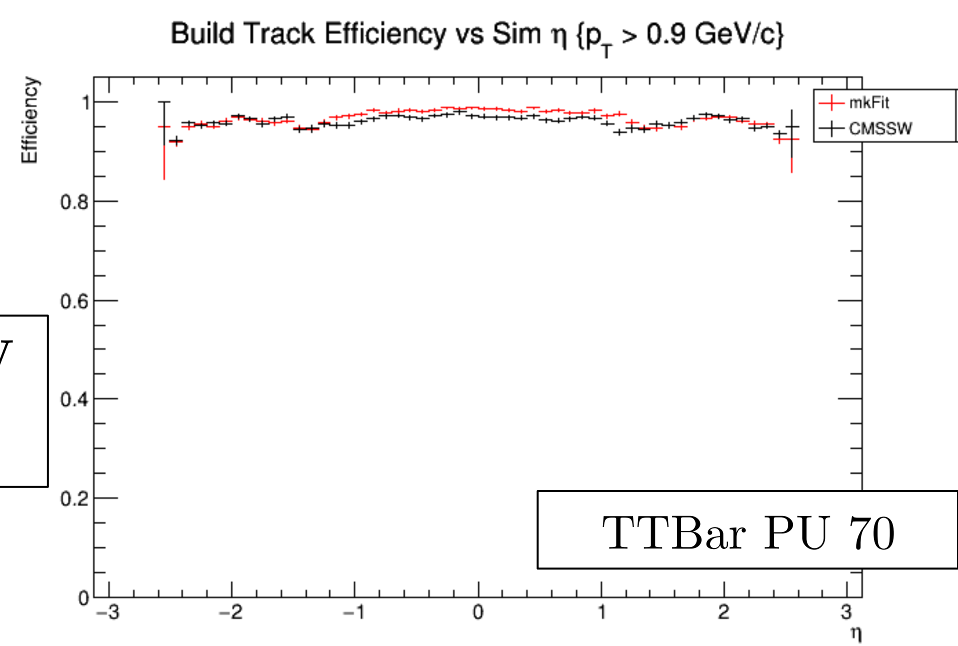


- To reduce combinatorics, CMS performs track finding over several iterations
  - Start with tracks that are easiest to find, end with the most difficult tracks
  - Between each iteration remove hits that have been associated to a track
- mkFit focuses on initial iteration:
  - Seed tracks with 4 hits and no beam-spot constraint
  - Find most prompt tracks
- Could easily be extended to include other iterations

Efficiency of mkFit

- Shown here: algorithm-level efficiency for long (≥ 12 hit) tracks
- **mkFit** is at least as efficient as **CMSSW, even for low $\mathbf{p_T}$ tracks**
  - Crucial for accurate particle flow reconstruction
- Much of the effort in the last year has focused on achieving this important milestone
- Next steps: improve efficiency for short tracks. Development for this is already in progress

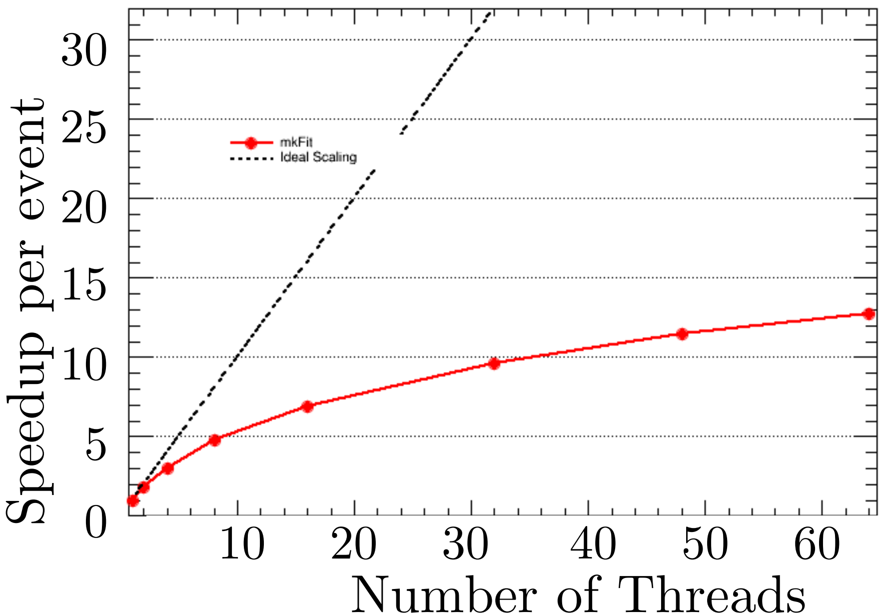




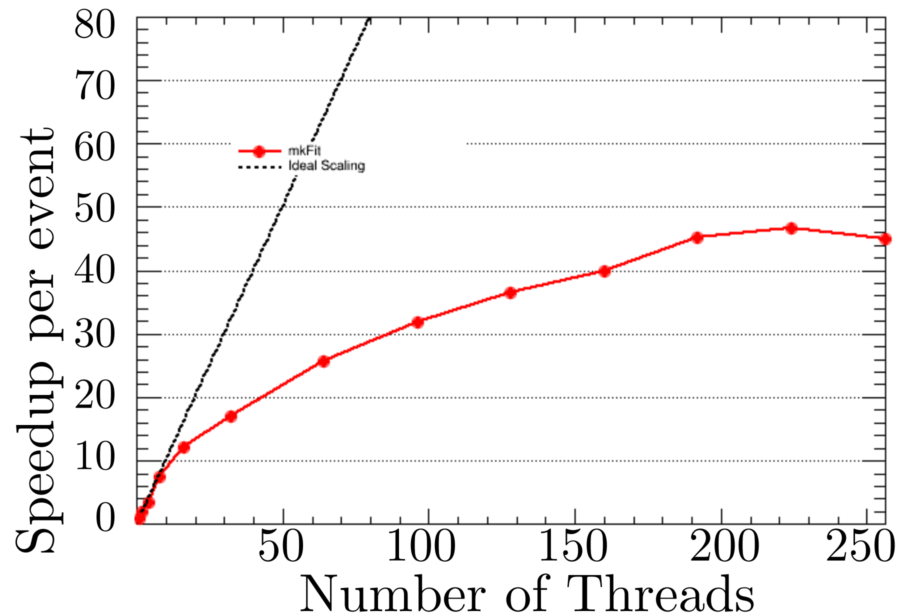
Speedup vs # of Threads

Excellent scaling at low threads –
independent of exact architecture

Intel Xeon
Skylake SKL (Gold 6130)



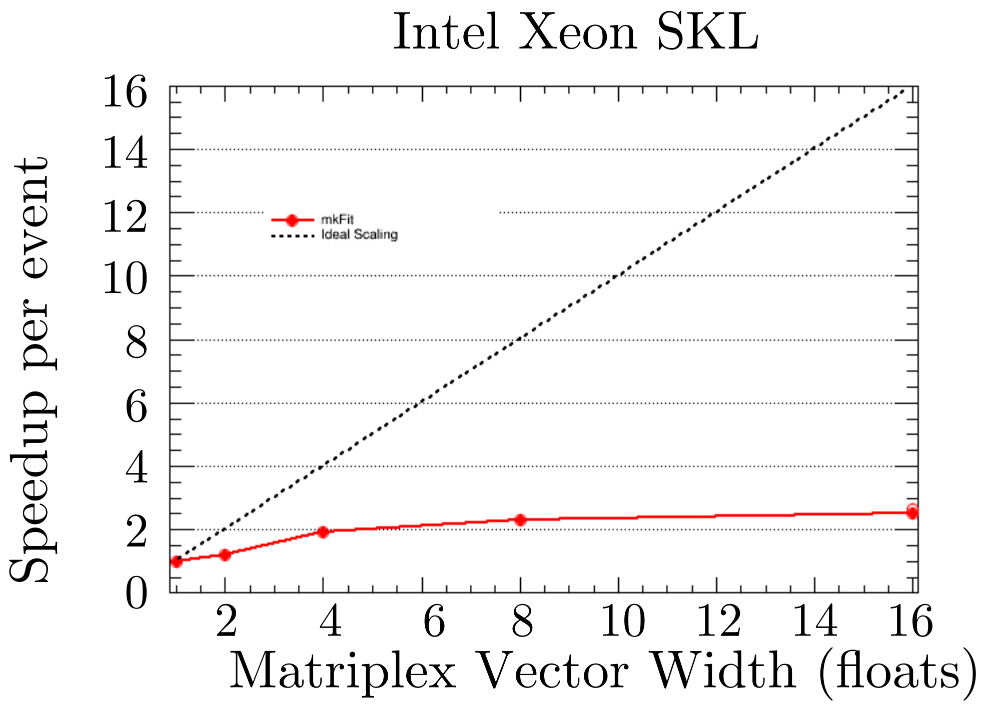
Intel Xeon Phi
Knight's Landing KNL (7210)



- Results for track building only; does not include overhead
- Measured using standalone configuration, single event in flight
- Turbo boost disabled

Speedup vs size of vector units

Algorithm uses vectorization successfully-
60 - 70% of code is vectorized!

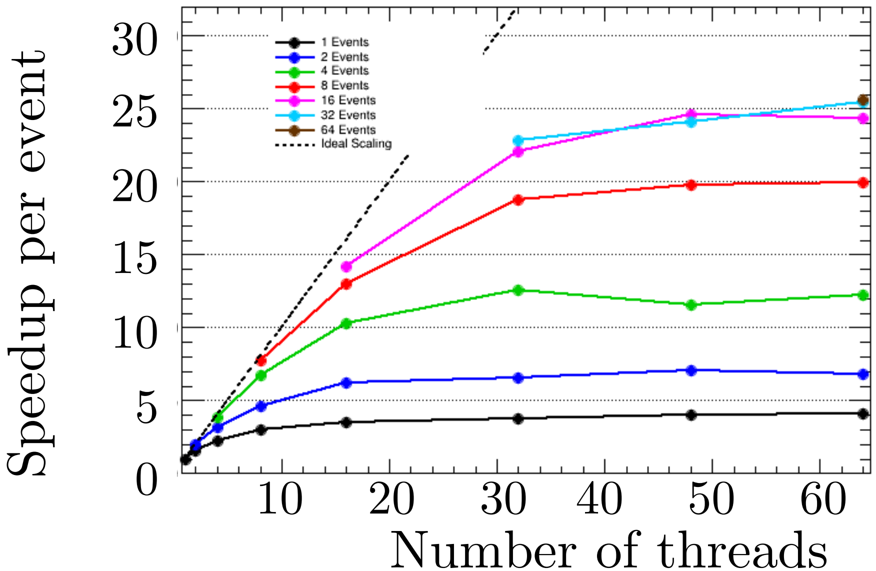


- Results for track building only; does not include overhead
- Measured using standalone configuration, single event in flight

Speedup vs # of events in flight

Can get speedups up to x25 using multiple events in flight

Intel Xeon SKL



- Results include time for full loop, including I/O, handling the seeds, etc
- Measured using standalone configuration
- Previous plots used only a single event in flight

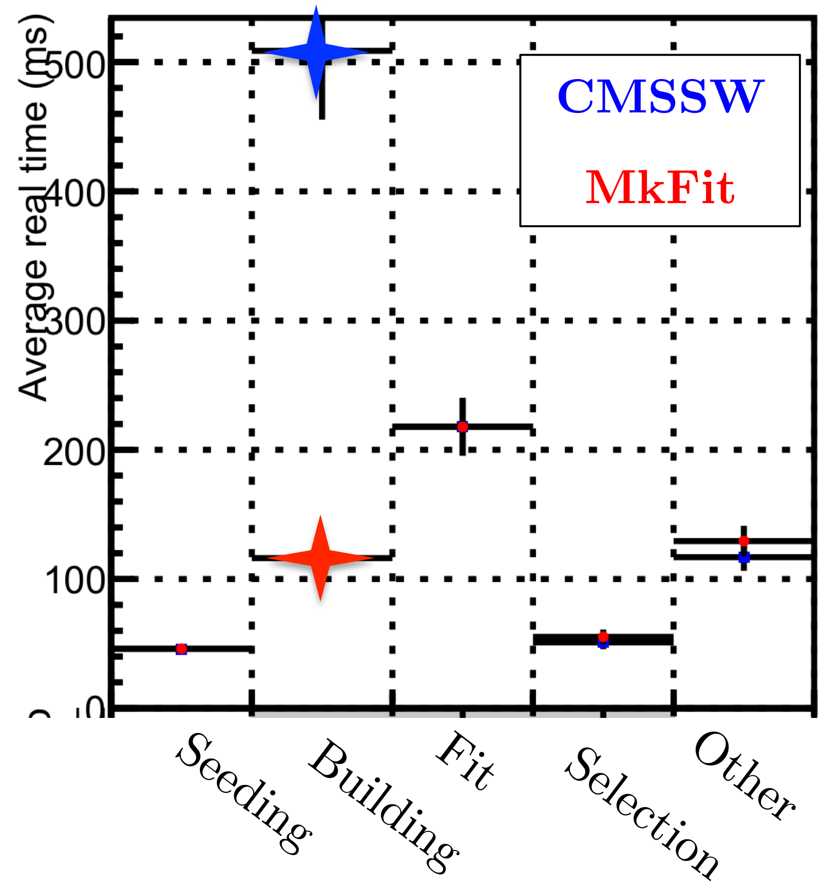
Integrated Timing Performance

Technical Details

- Run mkFit within CMSSW
- mkFit used for building only
- Single-thread test using TTBar PU 50

Results

- Track building is **4.3x faster**
- 40% of time is spent in data format conversions – actual track finding is **7x faster**
- Track building now takes **less time than track fitting**
- Even larger speedup if multiple threads are used



* Measured on SKL, mkFit compiled with AVX-512